VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

Analýza sériových protokolů

Serial Protocol Analyser

2010 Tomáš Kučera

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Tomáš Kučera**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Analýza sériových protokolů
Serial Protocol Analyser

Zásady pro vypracování:

Napište software pro analýzu sériové komunikace zaznamenané pomocí logického analyzátoru. Systém by měl být řešen modulárně, umožňujíc tak začlenění budoucích analytických modulů. Základní verze bude schopna graficky zobrazit zachycená data a umožnit tak vizuálně nastavovat parametry pro dekódování dat (možná je i automatická detekce rychlosti). Podporovány budou synchronní i asynchronní protokoly - RS232, PS/2 Mouse/Keyboard Protocol, I2C, 1-Wire, CAN. Program bude schopen dekódovaný bitový proud sestavit do odpovídající rámců (paketů) a ty uživateli zobrazit v čitelné formě.

1. Základní charakteristika sériových protokolů.
2. Analýza problému detekce bitové rychlosti v zaznamenaných datech.
3. Synchronizace, sestavování originálních bitových proudů.
4. Vizualizace, dekódování rámců.
5. Testování na reálných datech.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Krumnikl**

Datum zadání: 20.11.2009

Datum odevzdání: 07.05.2010

doc. Dr.Ing. Eduard Sojka
vedoucí katedry

prof. Ing. Ivo Vondrák, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal

Souhlasím se zveřejněním této bakalářské/diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských/magisterských programech VŠB-TU Ostrava.

V Ostravě dne 7.května 2010 .....

Na tomto místě bych chtěl poděkovat vedoucímu této práce Ing.Michalu Krumniklovi za trpělivost a pomoc při řešení problémů.

Abstrakt

Cílem této práce je vytvořit software pro analýzu sériové komunikace zaznamenané pomocí logického analyzátoru. První část se zabývá logickými analyzátory obecně a průzkumem dostupných řešení. Dále jsou popsány některé sériové protokoly. V druhé části je rozebrána implementace softwaru a jeho testování na reálných datech.

Klíčová slova:

Logický analyzátor, sériové rozhraní, bakalářská práce, RS232, I2C, CAN, PS/2, 1-Wire

Abstract

The aim of this work is to develop software for the analysis of serial communication recorded by logic analyzer. First part consider logic analyzers in general and research about available solutions. Also there are described several serial protocols. Software implementation and testing with real data is analyzed in second part.

Key words:

Logic analyzer, serial interface, bachelor thesis, RS232, I2C, CAN, PS/2, 1-Wire

Obsah

1. Úvod

Logický analyzátor slouží k číslicovým měřením. S jeho pomocí se zjišťují problémy při komunikaci mezi mikroprocesory, ladí číslicové obvody při vývoji atd. Zkrátka vše při čem potřebujeme sledovat více signálů, které mezi sebou mají nějakou časovou korelaci, najednou.

Logické analyzátory na rozdíl od osciloskopu nezkoumají analogový průběh signálu, ale zkoumají jen jeho logickou hodnotu, která je určena podle prahového napětí. Pokud je hodnota napětí na lince vyšší než je prahové napětí, logický analyzátor zaznamenává 1, pokud je napětí nižší analyzátor zaznamenává 0. Další markantní rozdíl oproti osciloskopu je v počtu vstupů. Dnešní logické analyzátory mohou mít stovky vstupů. Srdcem analyzátoru je akviziční paměť reálného času. Do této paměti se ukládají vzorkovaná data, musí tedy být dostatečně rychlá na to aby stihla vše zapisovat. U některých analyzátorů se data ukládají rovnou do PC, z některých se dají data po vzorkování z akviziční paměti do počítače stáhnout.

Zde přichází na řadu software pro zpracování takto získaných dat. Hlavní úlohou těchto programů je uživateli přehledně zobrazit průběhy signálů a tím usnadnit jejich zkoumání.

2. Software

Dnes je k většině dostupných logických analyzátorů vydáván i software. Po krátkém průzkumu člověk velice rychle zjistí, že mezi těmito programy jsou jen pramalé rozdíly, pokud zanedbáme GUI a i to bývá velice podobné. Základní a často jediná funkce programů je zobrazení několika průběhů. Nedílnou součástí ovládání je také možnost zoom, tedy nastavení vzdálenosti mezi sousedními vzorky. Tak aby se uživatel mohl rozhodnout zda chce raději vidět větší část přenesených dat nebo jestli potřebuje přesněji prozkoumat kratší úsek.

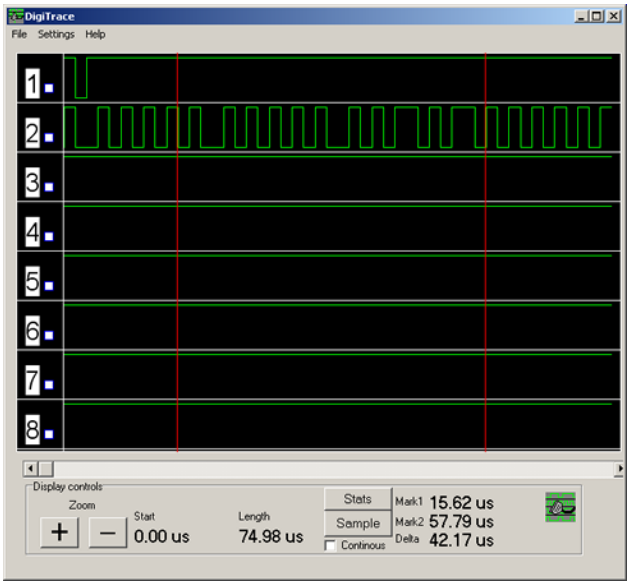
Na druhé straně spektra je mnohem menší skupina programů, které nabízejí spoustu velice zajímavých funkcí. Bohužel jejich hlavní nevýhodou je poměrně vysoká cena. Jako představitele této množiny programů bych zmínil software USBee suite pro, který je podle mého názoru velice silným nástrojem při analýze přenosů.

2.1 DigiTrace

DigiTrace slouží jako uživatelské prostředí k jednoduchému analyzátoru, který se k počítači připojuje přes paralelní port. Program je zdarma ke stažení z http://www.xs4all.nl/~jwasys/old/diy2.html .

Jde o domácí produkt a od toho se odvíjí i značně omezené schopnosti analyzátoru. Software data zachytává v reálném čase přímo ze sondy a vykresluje je na obrazovku. V případě potřeby se dá zapnout ukládání do souboru, aby se průběhy daly zkoumat i později. Bohužel maximální počet vzorků, které lze najednou získat je omezen na 32 768, což není mnoho. Obzvlášť když je využita maximální rychlost vzorkování 1 milion vzorků za sekundu. V nahraných datech se dá pohybovat a zvyšovat či snižovat množství zobrazených dat.

Tento jednoduchý program může být pro domácí potřeby či malé návrháře číslicových obvodů dobrým pomocníkem. Například při hledání chyb při synchronizaci obvodů. Jeho hlavní výhodou je snadná dostupnost a jednoduchost.



OBRÁZEK 1: OBRAZOVKA PROSTŘEDÍ PROGRAMU DIGITRACE, ZDROJ:[1]

2.2 USBee

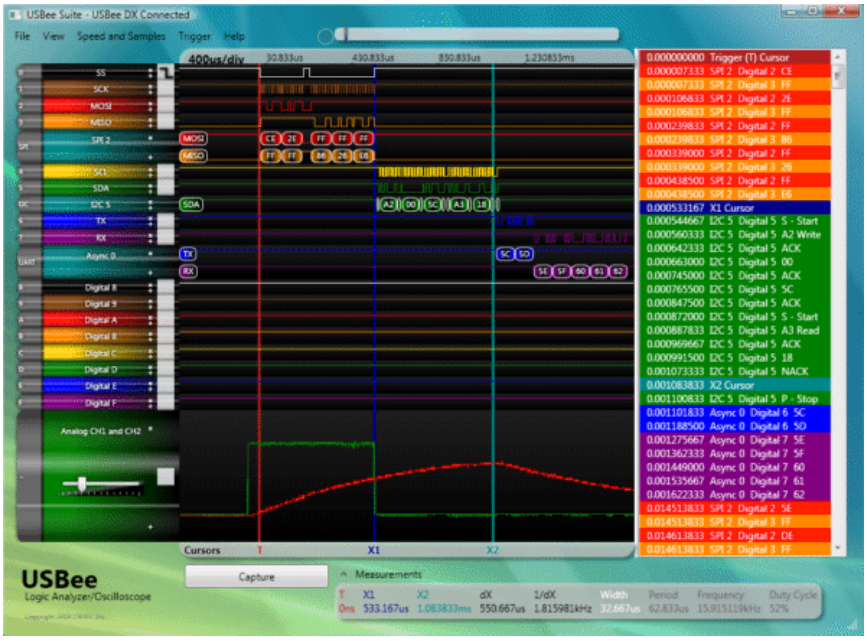
V rodině USBee je celá řada logických analyzátorů, které jak už název napovídá se k počítači připojují přes sériové rozhraní USB. Co je ovšem důležitější k těmto analyzátorům je možné si pořídit software USBee suite pro. Tento program se řadí mezi ty pokročilejší a nabízí uživateli řadu nástrojů a funkcí, které mu pomohou s analyzováním průběhů.

Už na první pohled zaujme velice příjemný vzhled aplikace, který se dá navíc do značné míry přizpůsobovat. V závislosti na použitém hardware se dá zobrazit osm i více linek. Tyto linky se dají kdykoliv ze zobrazení odstranit či přidat. Samozřejmě se každý průběh dá pojmenovat podle potřeby a dá se mu nastavit požadovaná barva.

Program podporuje hned několik druhů sběrnic jejichž rámce dokáže dekódovat a přehledně zobrazovat. Když chceme nějakou sběrnici přidat stačí kliknout na malé znaménko plus, které je v rohu u jména linky a pod tuto linku se vloží jeden prázdný pruh pro zobrazení dat. Zároveň se otevře dialog s výběrem různých rozhraní a dalšími nastaveními. Po přidání sběrnice se ve vloženém pruhu zobrazují rámce s vepsanými hodnotami. Zajímavým vylepšením je, že po najetí myší na nějaký rámec se tento jakoby zprůhlední a na jeho pozadí jde vidět původní signál, ze kterého byla data získána.

Dalším užitečným nástrojem je automatické měření periody signálu. Pokud kurzorem myši najedeme nad nějaký signál, zobrazí se v místě na které ukazujeme dvě malé kótovací šipky. Tyto šipky graficky označí zkoumanou periodu a v informačním poli se k této periodě vypíší vypočtené hodnoty délky periody, frekvence a poměru úseků log 1 a log 0.

Program dokáže zobrazovat i analogové signály a částečně tak nahradí i jednoduchý osciloskop.



OBRÁZEK 2: PROSTŘEDÍ PROGRAMU USBEE SUITE PRO, ZDROJ:[2]

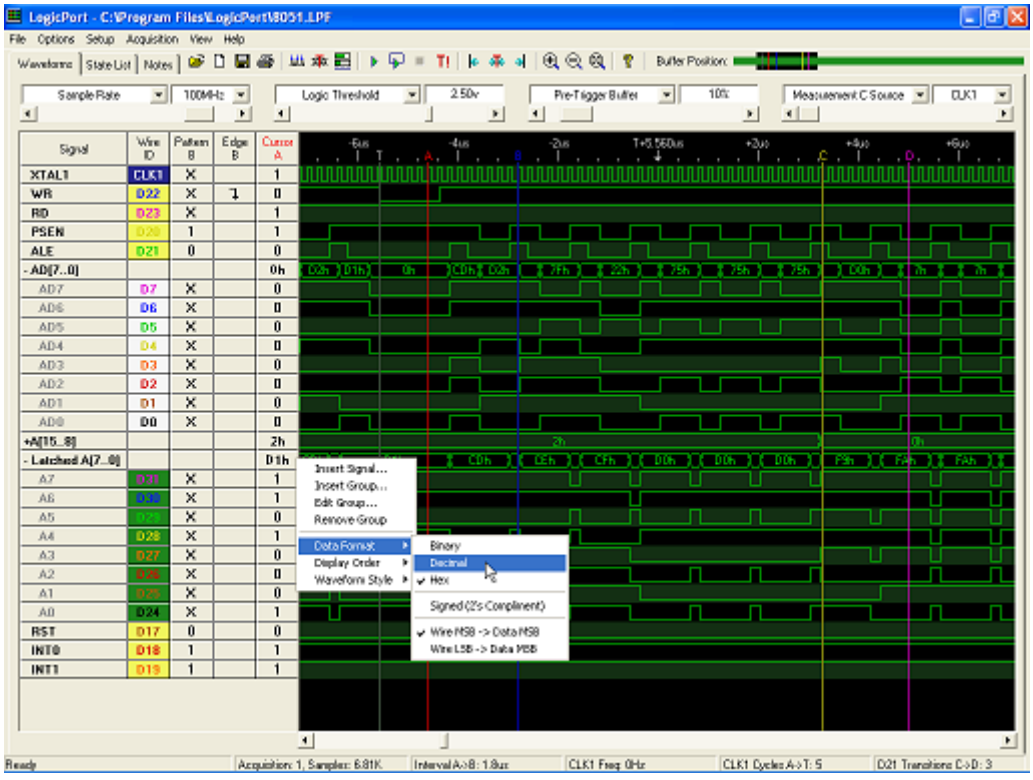
2.3 LA 1034 LogicPort

Tento logický analyzátor od firmy Intronix je střední cestou mezi drahými analyzátory s mnoha kanály a těmi levnými s omezenou vzorkovací frekvencí. LA 1034 nabízí 34 kanálů, které je možno vzorkovat s frekvencí 500MHz. Hardware je připojen k počítači přes port USB, tímto rozhraním je nejen ovládán ale také napájen.

V tomto případě je vzhled pojat více účelově, což nemusí být nutně na škodu. Při práci totiž nic neruší sledování průběhů.

Software analyzátoru podporuje několik druhů sběrnic např. I2C, CAN , SPI nebo RS 232. Sestavené rámce těchto sběrnic přehledně zobrazuje hned u signálů. Pro přidání sběrnice stačí kliknout pravým tlačítkem myší do tabulky se jmény signálů a sběrnic a zvolit možnost insert interpreter.
V zobrazeném okně se pak vybere požadované rozhraní a provedou se všechna nastavení.

Analyzátor dokáže také zachytit analogové průběhy pomocí A/D převodníku s vzorkovací rychlostí 100MHz. A tyto průběhy se pak mohou zobrazit u ostatních signálů.



OBRÁZEK 3: OBRAZOVKA PROGRAMU LOGIC PORT DODÁVANÉHO K ANALYZÁTORU LA 1034

2.4 Implementovaný software

V mé implementaci jsem se pokusil zpřístupnit některé funkce pokročilejších placených programů, tak aby bylo možno analyzovat základní protokoly.

Aplikace bude přijímat data z textových souborů generovaných softwarem, který je dodáván jako součást logického analyzátoru M611 od firmy ETC s.r.o. Tento analyzátor má 32 vstupních kanálů a dokáže snímat vzorky rychlostí až 100MHz. Maximální délka záznamu je 524 280 32-bitových slov. Pro připojení k PC jsou dvě možnosti. Buďto přes paralelní port nebo přes USB.

Analyzátor může pracovat ve dvou režimech. Prvním z nich je režim časové analýzy, ve kterém se využívá vnitřní zdroj taktovacích impulzů. V tomto režimu lze využít všech 32 kanálů. Druhou možností je režim stavové analýzy, kdy je jeden z kanálů využit jako vstup taktovacích impulzů a jeden kanál je určen pro zdroj povolovacího signálu. K připojení signálů slouží 4 sondy, každá s 8 vstupy.

	DigiTrace	USBee suite pro	LA 1034	Implementovaný software
zoom	•	•	•	•
měření času	•	•	•	•
změna barev/jmen		•	•	•
dekódování sběrnic		•	•	•
možnost zobrazení více sběrnic		•	•	
zobrazení analogových kanálů		•	•	
vyhledávání v datech		•	•	•

TABULKA 1: POROVNÁNÍ FUNKCÍ PROGRAMŮ

3.Protokoly

Analyzátor podporuje dekódování rámců těchto protokolů: I2C, PS2, RS232, 1-Wire. Protože všechny analyzované protokoly jsou sériové, bude pro naše potřeby stačit zobrazit osm průběhů. Jeden až dva pro signály sběrnic a zbytek pro možné sledování reakcí na dalších vodičích.

3.1 I2C

Sběrnice I2C neboli Internal-Integrated-Circuit Bus slouží především jako datová sběrnice pro propojení integrovaných obvodů - většinou uvnitř jednoho zařízení. Sběrnici vyvinula firma Philips a dnes se využívá především u různých mikrokontrolerů, DA a AD převodníků, sériových pamětí atd.

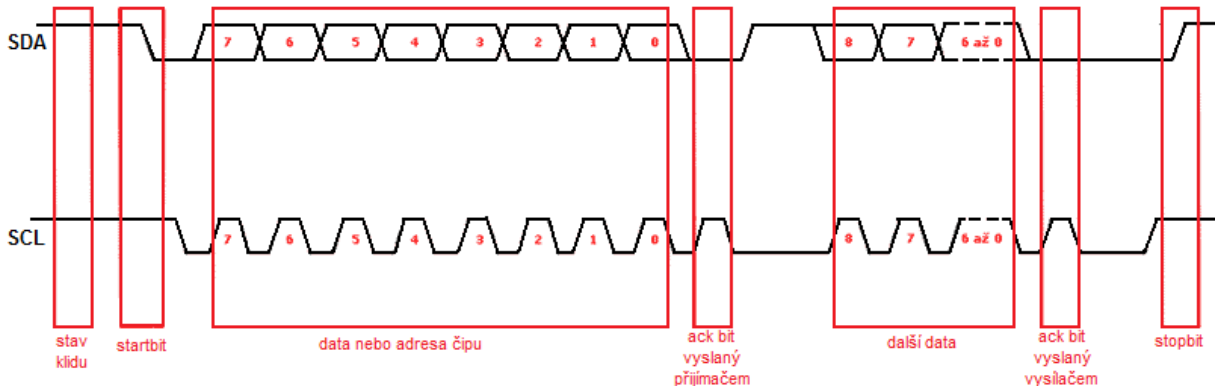
3.1.1 Popis

Veškerá komunikace probíhá na dvou vodičích, serial data (SDA) a serial clock (SCL). Na sběrnici může být připojeno značné množství zařízení, teoreticky až 1024, a to díky možnosti adresovaní čipů pomocí desetibitové adresy. V praxi je ale toto číslo mnohem menší. Z části také proto, že některé adresy jsou rezervovány pro zvláštní účely, např. broadcast, kdy naslouchají všechna zařízení připojená na sběrnici, ale hlavně proto, že většina čipů tak dlouhou adresu ani nepodporuje. Důležité je, že na sběrnici je vždy připojeno alespoň jedno zařízení typu master (typicky to bývá mikrokontroler), které se stará o generování hodinového signálu. Pokud nějaké zařízení vysílá, všechna ostatní naslouchají a podle adresy zjišťují, zda je komunikace určena jim či nikoliv.

3.1.2 Komunikace

Komunikace na sběrnici I2C je synchronní díky hodinovému signálu přenášenému po lince SCL. Protože pro data je určen jen jeden vodič SDA, je možná pouze komunikace poloduplexní.
V klidovém stavu je na obou linkách hodnota log. 1, což je zajištěno tím, že jsou obě linky přes pull-up rezistory připojeny k napájecímu napětí.

Pokud chce master komunikovat, stáhne linku SDA na log. 0, zatímco signál SCL ponechá nezměněn v log. 1 a tak inicializuje přenos. Poté začne master generovat hodinový signál a je vysílána adresa, kde poslední bit R/W určuje, zda se budou na adresované zařízení posílat data, nebo zda jsou od něj data očekávána. Log. 1 v bitu R/W znamená, že master bude data vysílat a v opačném případě bude vysílat slave. Zařízení, které rozpozná svou adresu poté potvrdí, že je připraveno tím, že linku SDA „stáhne“ na 0 po dobu jedné následující periody hodinového signálu Pokud je adresace potvrzena, vysílají se data 8 bitů od bitu s nejvyšší váhou (MSB) po bit s váhou nejnižší (LSB). Tento přenos musí příjemce dat opět potvrdit zasláním devátého bitu v hodnotě log. 0. Pak se mohou vysílat další data nebo je přenos ukončen příznakem stop - ten generuje master tím způsobem, že zatímco linka SLC zůstává v log. 1, linka SDA přejde z hodnoty 0 na hodnotu 1. Pokud není z nějakého důvodu přenos dat potvrzen, může master buď přenos ukončit, nebo se pokusí vyslat stejná data ještě jednou.



OBRÁZEK 4: PRŮBĚHY SIGNÁLŮ PŘI KOMUNIKACI NA SBĚRNICI I2C. ZDROJ:[8]

Každé zařízení má přiřazenu adresu (7 nebo 10 bitů), která jej na sběrnici jednoznačně identifikuje. Pokud se používá adresování 7-bitové, je adresa vysílána v prvním rámci celá společně s bitem RW. Pokud se používá adresování 10-bitové, vysílá se v prvním rámci speciální rezervovaná adresa (11110aa), kde jsou bity aa první dva bity adresy. V dalším rámci se pak vyšle zbývajících 8 adresních bitů.

Pro případ, kdy je na sběrnici více zařízení master, používá se ke komunikaci metoda s detekcí kolize. Metoda spočívá v tom, že zařízení vysílající bit zároveň kontroluje stav na lince SDA. Pokud zařízení vysílá log 1, ale na lince je přesto hodnota log. 0, musí zařízení přenos okamžitě ukončit, protože došlo ke kolizi.

3.2 RS232

Standard RS 232, který se původně používal k připojení terminálů k serveru se po krátké době stal poměrně oblíbeným a jeho použití se rozšířilo i do osobních počítačů a další elektroniky. V minulosti se prostřednictvím tohoto rozhraní připojovaly k PC například počítačové myši, dotykové LCD, některé druhy tiskáren, modemy, pomocí RS 232 se budovaly i jednodušší počítačové sítě.

3.2.1 Popis

Celkem má port rozhraní RS 232 devět pinů:

1. Data Carrier Detect (DCD)
2. Receive Data (RxD)
3. Transit Data (TxD)
4. Data Terminal Ready (DTR)
5. Grond
6. Data Set Ready (DSR)
7. Request To Send (RTS)
8. Clar To Send (CTS)
9. Ringing Indicator (RI)

Pro komunikaci ovšem postačují pouze tři z nich a to: RxD, TxD a Ground. Ostatní signály jsou spíše pomocné a mohou sloužit k řízení komunikace.

Při komunikaci přes port RS 232 můžeme v závislosti na potřebách využít simplexní, poloduplexní i plně duplexní způsob přenosu. Tento přenos probíhá vždy asynchronně, což lze poznat už podle toho, že se společně s daty nepřenáší žádný hodinový signál.

Pokud není potřeba plně duplexní přenos, ale jen poloduplexní nebo simplexní, stačí pro propojení zařízení jen dva vodiče. Jeden pro data, druhý pro společnou signálovou zem. Pro přenos plně duplexní jsou potřeba minimálně tři vodiče. Jeden pro společnou zem (Ground), jeden pro vysílání dat (TxD) a jeden pro příjem dat(RxD). Na zařízeních se vodiče musí překřížit, tak že je TxD strany A spojen s RxD strany B a obráceně RxD strany A spojen s TxD strany B.

3.2.2 Komunikace

Jak už jsem zmínil, komunikace po sběrnici RS232 probíhá asynchronně, to znamená, že hodinový signál si musí každá stanice generovat sama. To ovšem klade značné nároky na přesnost generátoru hodinového signálu. Pokud by nedocházelo k žádné synchronizaci, rozdíl v délce jednotlivých period hodinových signálů by se neustále sčítal. V praxi by to znamenalo, že již po pár vteřinách přenosu by byla komunikace nemožná. Proto se data vysílají po rámcích (5 – 8 bitů), přičemž na začátek i konec těchto rámců se vkládají synchronizační značky. Na začátku rámce je to tzv. start bit a na jeho konci zase stop bit.

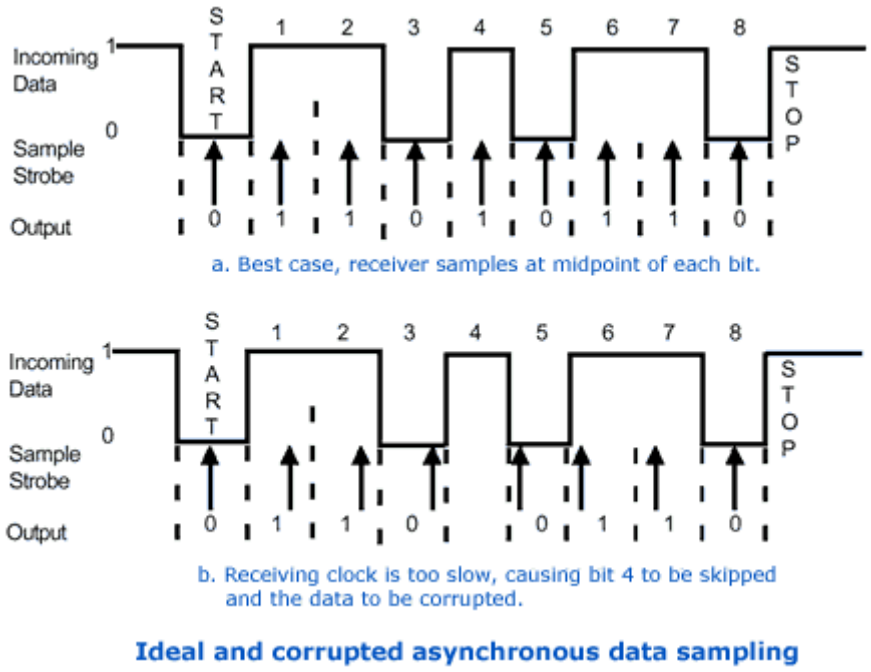
Dále je možno do rámce vložit jeden či více paritních bitů pro základní detekci chyb. Star bit má vždy hodnotu log. 0 zatímco stop bit má hodnotu opačnou log. 1. Tímto je zajištěno, že i kdyby všechny datové bity v rámci měly stejnou hodnotu, dojde na sběrnici ke změně stavu a tím k synchronizaci přenosu.

U zařízení na obou stranách sběrnice je potřeba nakonfigurovat:

1. Rychlost přenosu tak, aby obě strany věděly jak rychle vysílat (vzorkovat) data.
2. Počet přenášených datových bitů v jednom rámci.
3. Počet paritních bitů.
4. Délku stopbitu, který může být delší než bity ostatní.

Důvodem k delšímu stopbitu je zajištění dostatečného času pro přijímací stranu, aby mohla přenesená data zpracovat, než se začnou vysílat data nová.

Na obrázku 5 lze vidět příklad přenosu po sběrnici RS232. Na sběrnici v klidovém stavu je udržována hodnota log. 1. Začátek přenosu oznámí vysílač, tím že vyšle start bit (stáhne hodnotu na log. 0), poté je vysláno 8 datových bitů a nakonec se sběrnice vrací na hodnotu log. 1, což ukončuje přenos rámce.



OBRÁZEK 5: UKÁZKA PŘENOSU JEDNOHO BAJTU PO SÉRIOVÉ LINCE. NAHOŘE IDEÁLNÍ STAV, DOLE DOCHÁZÍ K CHYBĚ, ZDROJ:[5]

3.3 1-Wire™

Sběrnice 1-Wire™ byla vyvinuta firmou Dallas Semiconductor a obyčejně se používá k připojení digitálních teploměrů a jiných levných zařízení k mikroprocesoru. Na této sběrnici je postavená také technologie iButton.

3.3.1 Popis

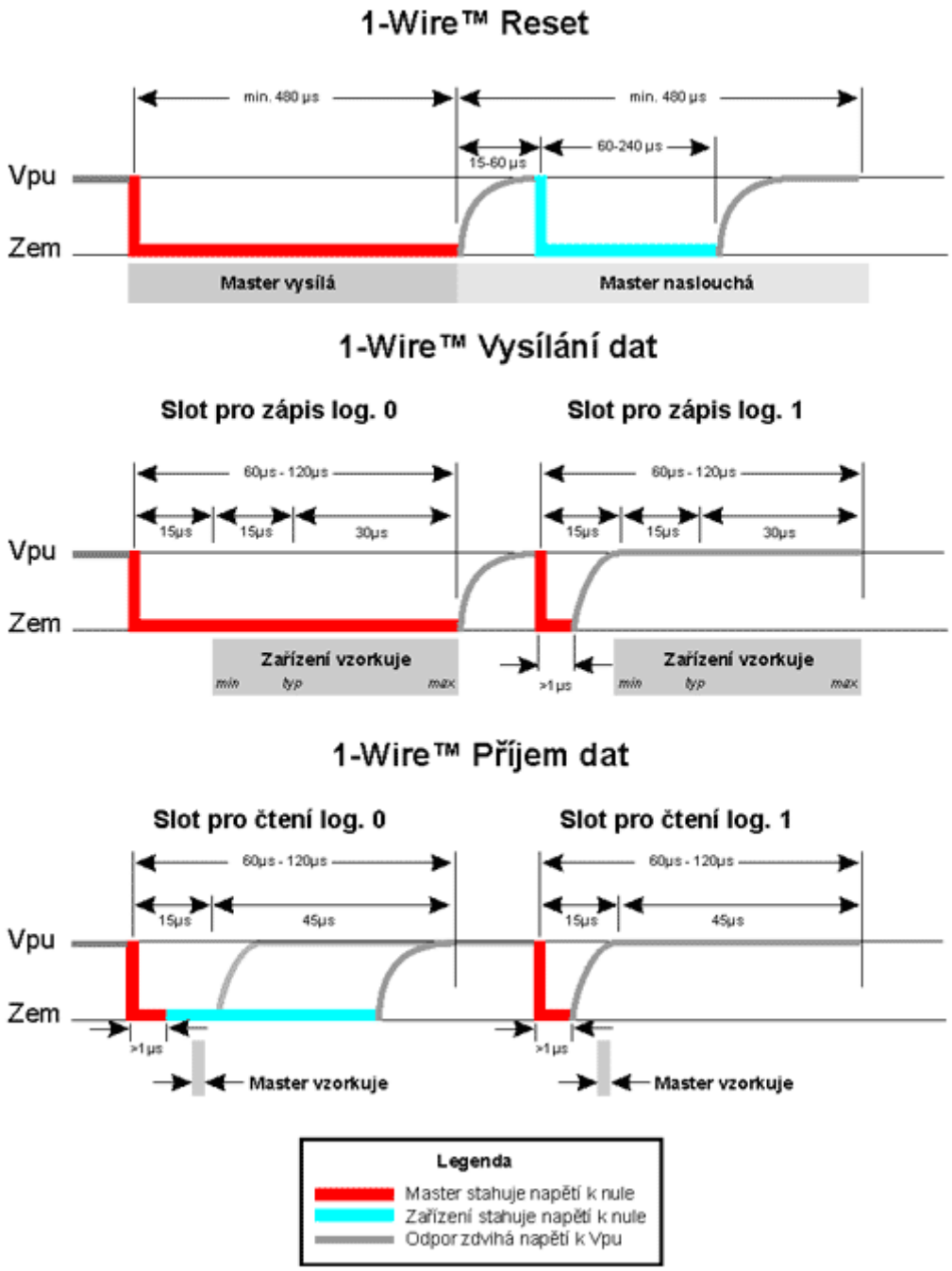
K sběrnici je vždy připojeno jedno zařízení master (obyčejně mikroprocesor) a jedno či více zařízení slave. K propojení stačí dva vodiče, a to společná zem a datový vodič. Na datovém vodiči je standardně hodnota log. 1, díky tomu, že je připojen přes rezistor k napájecímu napětí.

Pokud je na sběrnici připojeno více zařízení 1-Wire™, lze je jednoznačně identifikovat podle unikátního 64-bitového kódu, který je tvořen z 8 bitů pro typ zařízení, 48 bitů sériového čísla a dalších 8 bitů CRC kódu.

3.3.2 Komunikace

Komunikace po sběrnici 1-Wire™ je poloduplexní a asynchronní, zahajuje ji vždy zařízení master, které vyšle tzv. reset pulz. Ten probíhá tak, že master uzemní datový vodič na dobu minimálně 480µs. Poté sběrnici uvolní a ta se díky pull-up rezistoru vrátí na hodnotu log. 1. Pokud je na sběrnici připojeno nějaké 1-Wire™ zařízení, zachytí tuto vzestupnou hranu a ohlásí se obdobným způsobem, po prodlevě (15 – 60µs) sběrnici uzemní na 60-240µs. Po tom, co se nějaké zařízení ohlásí, může začít samotný přenos. Jednotlivé bity jsou přenášeny v tzv. „time-slotech“. Time slot začíná vždy master tím, že uzemní sběrnici na 1µs. Další postup se liší podle toho, jestli master data čte nebo vysílá. Pokud master vysílá, nechá sběrnici uzemněnou (log. 0), popřípadě ji uvolní (log. 1), toto trvá 60-

120µs. Když master data přijímá, stáhne sběrnici k log. 0 na 1µs a poté ji uvolní. Slave může vyslat bit tak, že sběrnici sám uzemní nebo ji ponechá ve stavu log. 1. Mezi jednotlivými time-sloty musí být vždy mezera alespoň 1µs.



OBRÁZEK 6: UKÁZKY PRŮBĚHŮ STAVŮ NA DATOVÉ LINE, PŘI RESETU A RÁMCÍCH PRO PŘÍJEM I VYSÍLÁNÍ DAT, ZDROJ:[7]

3.4 PS/2

Sběrnice PS/2 neboli (Personal System/2),vyvinutá firmou IBM, se objevila někdy v osmdesátých letech a dodnes se s ní můžeme setkat u připojování vstupních zařízení k PC. Jedná se hlavně o myš a klávesnici, i když v poslední době toto rozhraní nahrazuje novější USB.

3.4.1 Popis

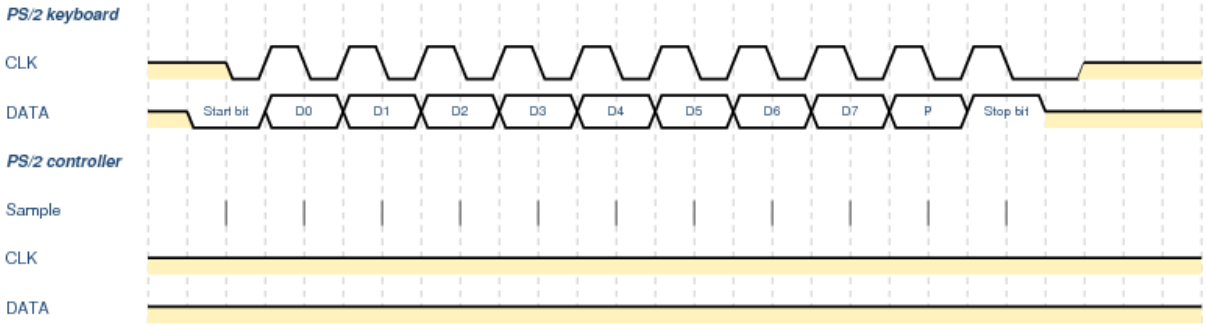
PS/2 je synchronní obousměrný protokol typu master-slave. K jednomu zařízení master je vždy na druhém konci připojeno pouze jedno zařízení slave. Specifikem protokolu je to, že hodinový signál většinou generuje koncové zařízení a nikoliv počítač jak bychom očekávali.

Přestože se pro připojování používal 5-pinový DIN nebo 6-pinový mini-DIN konektor, tvoří rozhraní jen 4 vodiče: napájení, zem, data a clock.

3.4.2 Komunikace

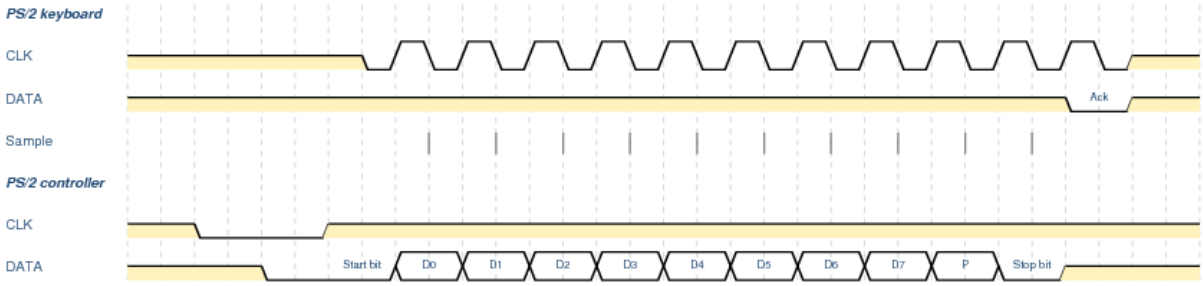
Linky data i clock jsou připojeny ke zvedacím rezistorům, takže v klidovém stavu je na obou log. 1, připojení je řešeno pomocí otevřeného kolektoru. Přenosový rámec má jedenáct bitů, skládá se z startbitu, jednoho bajtu dat, paritního bitu a stop bitu.

Komunikace od koncového zařízení může být zahájena pokud obě linky(data i clk) byly na úrovni high alespoň 50 µs. Přenos začíná start bitem na úrovni low, po přenesení dat a parity je přenos ukončen stop bitem s úrovní high.



OBRÁZEK 7: PRŮBĚH PŘENOSU Z KONCOVÉHO ZAŘÍZENÍ (ČTENÍ). ZDROJ:[10]

Přenos opačným směrem je trochu odlišný. Inicializuje jej řadič stažením datového vodiče na log.0. Před tím než řadič toto provede, musí se postarat o to aby zařízení nezačalo přenos opačným směrem. Toho docílí tak, že na dobu minimálně 100 µs stáhne clock na log.0. Po vyslání start bitu řadič linku s hodinami zase uvolní a počká až zařízení začne generovat hodinový signál. Následuje přenos jednoho bajtu dat a parity, ten je ukončen stop bitem. Aby byl přenos zdárně ukončen musí ještě zařízení vše potvrdit bitem acknowledge, tak že data stáhne na úroveň low.



OBRÁZEK 8: PRŮBĚH PŘENOSU DO KONCOVÉHO ZAŘÍZENÍ (ZÁPIS). ZDROJ:[10]

3.5 CAN

Sběrnice CAN se používá k propojování mikropočítačů, čidel a akčních členů. Nejčastěji se s ní setkáme v průmyslu a v automobilech.

3.5.1 Popis

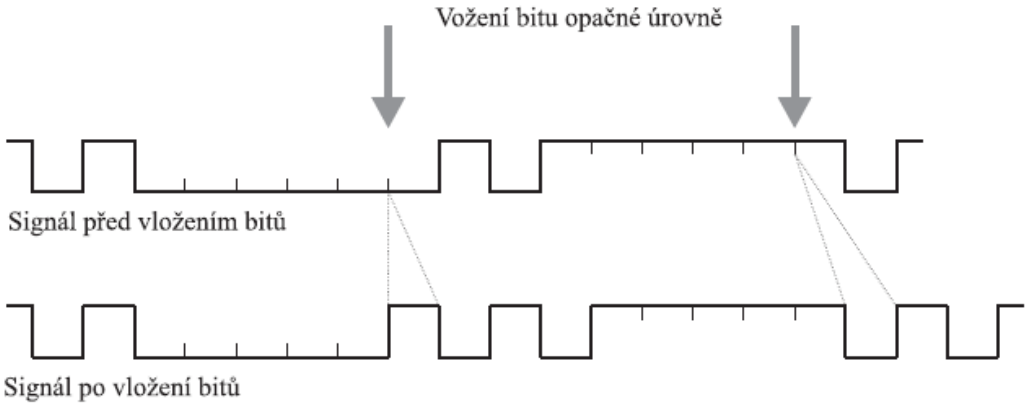
Jednotlivá zařízení vysílají data nezávisle na tom jestli o ně někdo žádal či nikoliv. Datové rámce proto nejsou opatřeny adresou cílové stanice, nýbrž identifikátorem nesených dat. Rámce tedy zachycují všechna zařízení a až na základě identifikace se rozhodují jestli daný rámec budou zpracovávat dále nebo jestli jej vymažou. Existují dva druhy rámců. Jeden s 11 bitovým identifikátorem a poté rozšířený rámec s 29 bitovým identifikátorem.

Sběrnice je realizována jediným vodičem. Na lince se rozlišují dva stavy dominantní a recesivní. Mezi těmito stavy existuje tento vztah: pokud všechna připojená zařízení vysílají recesivní úroveň výsledný stav na lince je také recesivní. Pokud ale alespoň jedno zařízení vysílá dominantní úroveň bude také na lince dominantní úroveň. Tento mechanismus je důležitý pro arbitraci sběrnice, kterou popíši později.

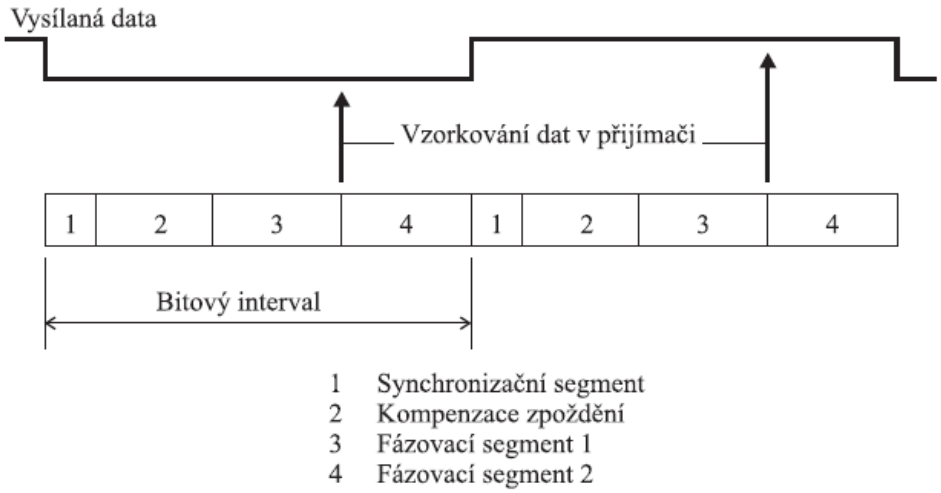
Protože zde není k dispozici žádný vodič pro synchronizaci, generují si všechna zařízení vlastní hodinový signál, který se synchronizuje pomocí datové linky. Aby byla synchronizace dostatečně efektivní, používá se metoda vkládaných bitů. Pokud jde bezprostředně po sobě pět bitů se stejnou hodnotou, je za ně vložen jeden bit s hodnotou opačnou. Tento vložený bit je pak na straně příjemce z dat automaticky odstraněn. Tímto způsobem je zaručeno, že vždy po maximálně pěti bitech dojde k synchronizaci hodin.

Přenos každého bitu se skládá ze 4 částí, jak je vidět na obrázku 10.

Pokud dojde ke změně stavu na lince jindy než v první fázi, znamená to, že se hodiny na vysílací a přijímací straně rozcházejí a je potřeba provést korekci.



OBRÁZEK 9: VKLÁDÁNÍ SYNCHRONIZAČNÍCH BITŮ, ZDROJ[3]



OBRÁZEK 10: PŘENOS BITŮ A JEHO ČÁSTI, ZDROJ[3]

3.5.2 Komunikace

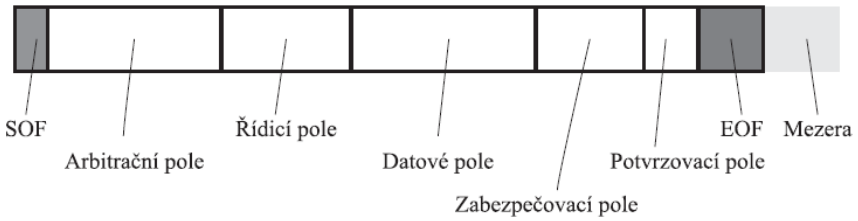
Přenosy po sběrnici probíhají po rámcích. Existují 4 druhy rámců. Obyčejný datový rámec, CAN Remote Frame, který je vysílán jako požadavek na data a dva druhy chybových rámců.

Standardní rámec, který je vidět na Obrázku 11 se skládá z těchto sedmi částí:

1. Start bit SOF, má vždy dominantní úroveň.
2. Arbitrační pole obsahuje 11 bitů identifikátoru + bit RTR, který určuje zda se jedná o rámec s daty nebo o požadavek na data.
3. Dalším je řídící pole v němž se vysílá bit IDE. Podle bitu IDE se rozpozná zda jde o standardní či rozšířený datový rámec. V tomto případě má dominantní hodnotu. Dále je zde bit r0, tento bit je rezervován pro pozdější využití a je vždy dominantní. Jako poslední jsou čtyři bity DLC určující délku datového pole(0-8 bitů).
4. V datovém poli se přenáší samotná data, má délku závislou na hodnotě DLC

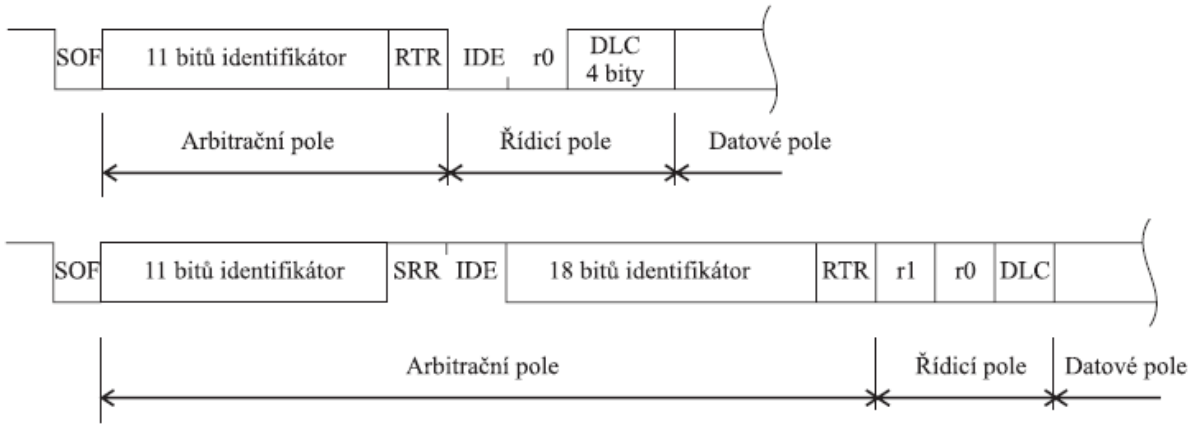
5. V zabezpečovacím poli je přenášen 15bitový CRC kód, který se počítá ze všech předcházejících polí, včetně SOF. Poslední bit tohoto úseku je zakončovaní.
6. Potvrzovací pole obsahuje pouze dva bity. ACK a zakončovaní bit. Vysílač oba tyto bity vysílá v recesivní úrovni. Přijímač, který data zpracovává vyšle oba tyto bity jako dominantní, tím se vysílací strana dozví o tom, že její data někdo akceptuje.
7. Poslední zakončovaní pole EOF se skládá ze 7 bitů, všechny mají recesivní úroveň.

Mezi jednotlivými rámci je ještě nutné nechat mezeru alespoň 3 bity, kdy je na lince recesivní úroveň.



OBRÁZEK 11: STANDARDNÍ RÁMEC CAN S 11 BITOVOU IDENTIFIKACÍ, ZDROJ[3]

Na dalším obrázku lze vidět rozdíl ve složení standardního a rozšířeného rámce. V rozšířeném rámci je navíc bit SRR, který se vysílá vždy v recesivní úrovni. Bit IDE má také hodnotu recesivní a tím indikuje že se jedná o rozšířený rámec. Následujících 18 bitů je druhá část identifikace. Nakonec k rezervovanému bitu r0 přibyl ještě r1, oba se vysílají jako dominantní.



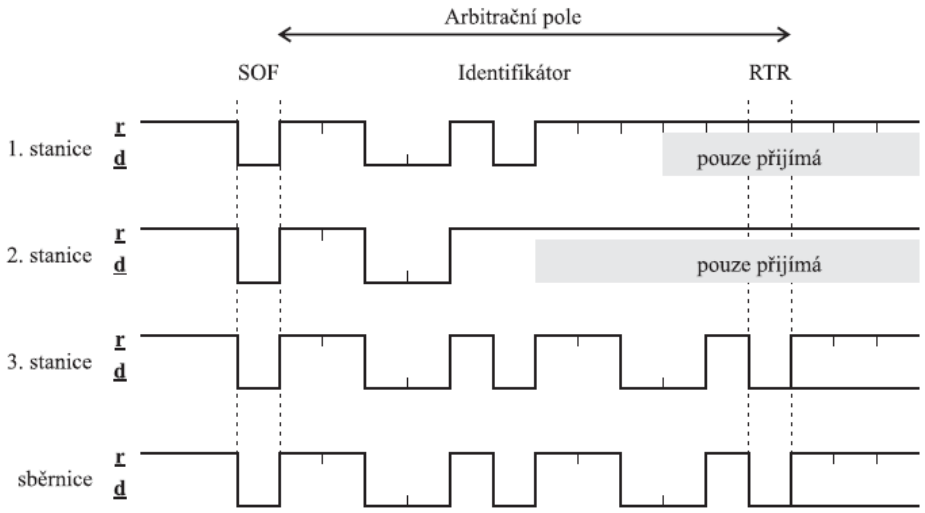
OBRÁZEK 12: ROZDÍLY MEZI STANDARDNÍM A ROZŠÍŘENÝM RÁMCEM, ZDROJ[3]

Rámec s požadavkem na vysílání dat je velice podobný datovému rámci. Změna je v arbitračním poli v bitu RTR (Remote transfer Request), ten se vysílá v úrovni recesivní. Datové pole, tohoto rámce má nulovou délku.

3.5.3 Arbitrace

Arbitrace sběrnice CAN vychází z toho, že současně může začít vysílat více stanic a využívá vztah, který je mezi recesivní a dominantní úrovní. Každá stanice totiž při zápisu na sběrnici zároveň kontroluje stav linky. Pokud je stav shodný se zapisovanou hodnotou je vše v pořádku. Pokud ale zařízení zapisuje úroveň recesivní a na lince je přesto úroveň dominantní, znamená to, že vysílá i někdo jiný. Když stanice takový stav detekuje, okamžitě přechází do přijímacího stavu.

Tento mechanismus spolu s pravidly pro bit RTR zajišťuje přednost datových rámců před rámci s požadavkem. Pokud totiž začne stanice požadující data vysílat ve stejném čase jako stanice stejná data poskytující, u bitu RTR dojde ke kolizi. Protože v datovém rámci je RTR v dominantní úrovni má tento rámec přednost a stanice, která původně chtěla vyslat požadavek přejde do režimu pro příjem dat a rámec zachytí.



OBRÁZEK 13: UKÁZKA SOUBĚŽNÉHO POKUS 3 STANIC O ZÁPIS NA SBĚRNICI, ZDROJ[3]

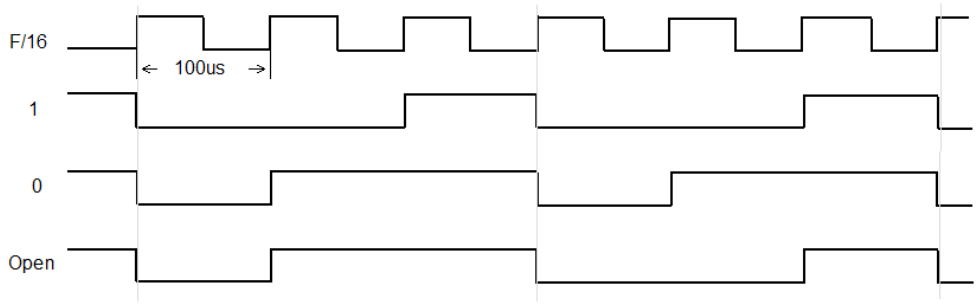
3.6 UM3758

Integrované obvody série UM3758 od firmy United Microelectrinic corp. se používají pro přenos dat s adresací dlouhou až 18 bitů. Každý integrovaný obvod má několik vstupů sloužících pro elektronické nastavení adresy.

3.6.1 Popis

Komunikace mezi jednotlivými obvody je asynchronní. Obvod má vlastní generátor hodinového signálu. Proto se před každým rámcem vysílá synchronizační pulz. Pro přenos bitů se používá tří stavový kód, ten je využit u adresování. Takže maximální počet různých adres je 18^3 tedy více než 387 miliónů kombinací.

Na obrázku lze vidět jak jsou jednotlivé hodnoty zakódovány. Log. 1 se vysílá jako dva krátké pulzy. Log. 0 jako dva dlouhé pulzy a hodnota open se vysílá jako jeden dlouhý a jeden krátký pulz.



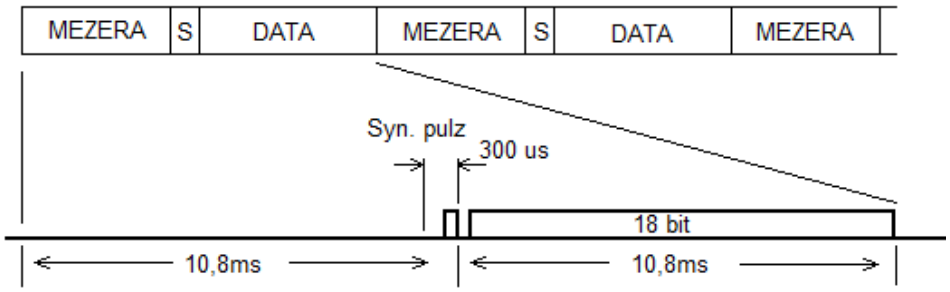
OBRÁZEK 14: TŘÍ STAVOVÉ KÓDOVÁNÍ, ZDROJ[11]

Pokud jde o data tak se bit open počítá jako log. 1.

3.6.2 Komunikace

Když je obvod ve vysílacím režimu vysílá neustále adresu + data. Obvody naslouchající na sběrnici porovnávají adresu se svou vlastní adresou a pokud se shoduje ukládají si data do vnitřní paměti. Při následujícím čtení se opět porovná adresa a s ní se porovnávají i přijatá data s daty uloženými z předchozího rámce. Pokud se podaří přijmout třikrát po sobě stejná data, jsou tato prohlášena za platná a obvod je nastaví na své výstupní piny. Tyto hodnoty zůstanou na výstupu do té doby než se přijmou nová validní data.

Mezi jednotlivými rámci musí být na sběrnici mezera. Na následujícím obrázku lze vidět časový harmonogram vysílání pro obvod s 10 adresními bity a 8 bity dat.



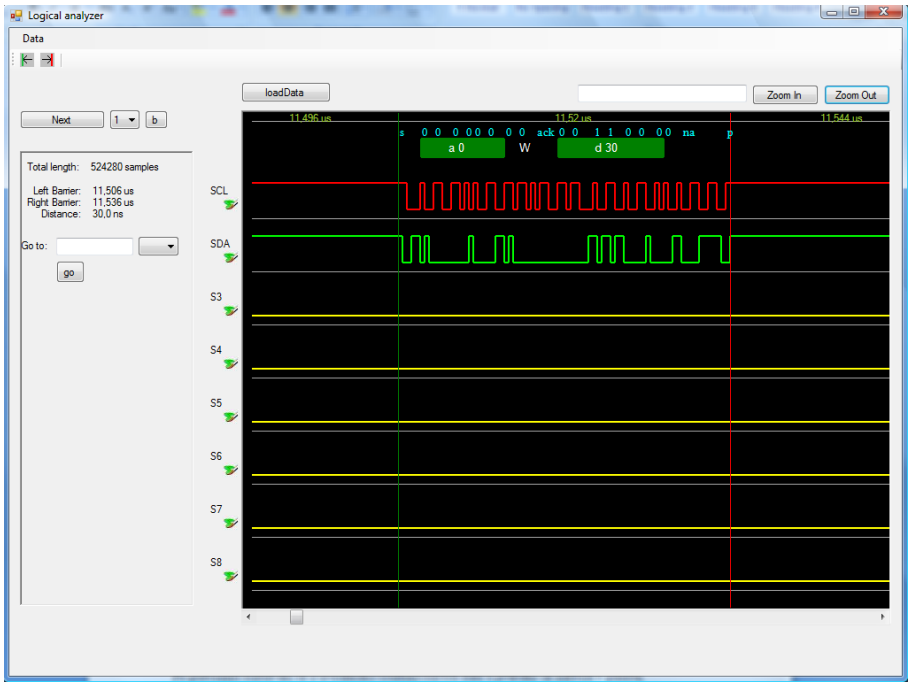
OBRÁZEK 15: PŘENOS RÁMCŮ PRO DEKODÉR UM3758

4. Implementace

Pro implementaci jsem zvolil framework .NET 3.5, který mne oslovil svou komplexností a množstvím podpůrných knihoven. Díky překladu kódu do MSIL(Microsoft Intermediate Language), který se do strojového kódu překládá až na cílové platformě by měl být software i přenositelný mezi operačními systémy. Pro běh v operačních systémech Unixového typu lze použít běhové prostředí Mono či DotGNU Portable.NET.

4.1 Popis funkcí

Po spuštění programu uživatel vidí základní okno aplikace zobrazené na Obrázku 15. V podstatě všechny funkce jsou dostupné z tohoto okna.



OBRÁZEK 16: PROSTŘEDÍ IMPLEMENTOVANÉHO ANALYZÁTORU

4.1.1 Načítání dat

Pravděpodobně nejdůležitější funkce je načtení navzorkovaných dat. Program podporuje textové soubory exportované z software, který je dodáván společně s logickým analyzátorem.

Po kliknutí na tlačítko Load Data se otevře dialogové okno pro výběr souboru. Když je soubor vybrán a potvrzen objeví se další okno tentokrát pro nastavení zobrazení. Uživatel má možnost každému z osmi signálů vybrat barvu, nastavit jméno a rozhodnout zda se tento signál vůbec zobrazí či nikoliv. Dalším poměrně důležitým nastavením je zde rychlost vzorkování s jakou byla data získána. Toto má vliv na údaje na časové ose, důležitější je ovšem vliv tohoto nastavení na dekódování některých sběrnic závislých na čase.

Po potvrzení nastavení se z uvedeného souboru načtou data a průběhy se zobrazí v panelu.

4.1.2 Panel se signály

Po zobrazení dat v panelu se aktualizují názvy signálů, které jsou uvedeny vždy vedle dané linky. Pokud by chtěl uživatel toto pojmenování dodatečně změnit, stačí kliknout na jméno a do zobrazeného okna zadat nový název. Změnit se dá i barva signálu. Kliknutím na ikonku štětce, která je umístěna těsně pod názvem, se spustí dialog pro výběr barvy.

Pomocí tlačítek Zoom In a Zoom Out se dá zvyšovat/snižovat vzdálenost mezi jednotlivými vzorky a tím nastavovat množství najednou zobrazených vzorků.

4.1.3 Měření času

Někdy se může hodit funkce pro změření určitého úseku dat, například doba přenosu jednoho rámce nebo čas který uplynul mezi jednotlivými vysíláními.

K tomuto účelu slouží tzv. „zarážky“. Zarážky jsou dvě, levá a pravá. Jejich ikony jsou umístěny v pravém horním rohu. Když chce uživatel zarážku umístit, prostě stačí kliknout na její ikonu. Poté po najetí myší nad panel se signály se začne vykreslovat svislá čára naznačující kam se zarážka umístí. Zároveň se v informačním panelu aktualizuje pozice zarážky. Pokud jsou umístěny obě zarážky vypočte se časový rozdíl, který se dá vyčíst v kolonce distance.

4.1.4 Rychlý přechod na pozici

Protože pohybovat se v datech pouze pomocí horizontálního posuvníku může být dosti nepřesné, hlavně v případech kdy je načteno větší množství vzorků. Jsou zde dvě možnosti jak zobrazit požadovaný úsek.

První možností je v informačním panelu vyplnit požadovaný čas do políčka Go to, následně vybrat jednotku ve které je čas udán a zmáčknout tlačítko Go. Tím se zobrazí část průběhů jejíž čas začíná na požadované hodnotě.

Alternativou je vyhledávání výskytu změn v datech. Často se může stát, že mezi relevantními úseky jsou dlouhé mezery kdy je sběrnice v klidovém stavu. Pro vyhledávání takových částí slouží tlačítko Next. Z přilehlého select boxu si uživatel určí, kterou linku by chtěl prohledat. Kliknutím na Next se najde první změna hodnoty na určené lince. Prohledávají se pouze data následující vzhledem k aktuální pozici, takže po prostudování se dá vyhledat další výskyt změn.

4.1.5 Dekódování sběrnic

Když jsou data načtena, je možnost nechat analyzátor aby se pokusil sběrnici dekódovat.

Dialog pro přidání sběrnice je spouštěn možností Add Bus. Ze záložek se pak vybere požadovaná sběrnice. Každá sběrnice má pak svá nastavení.

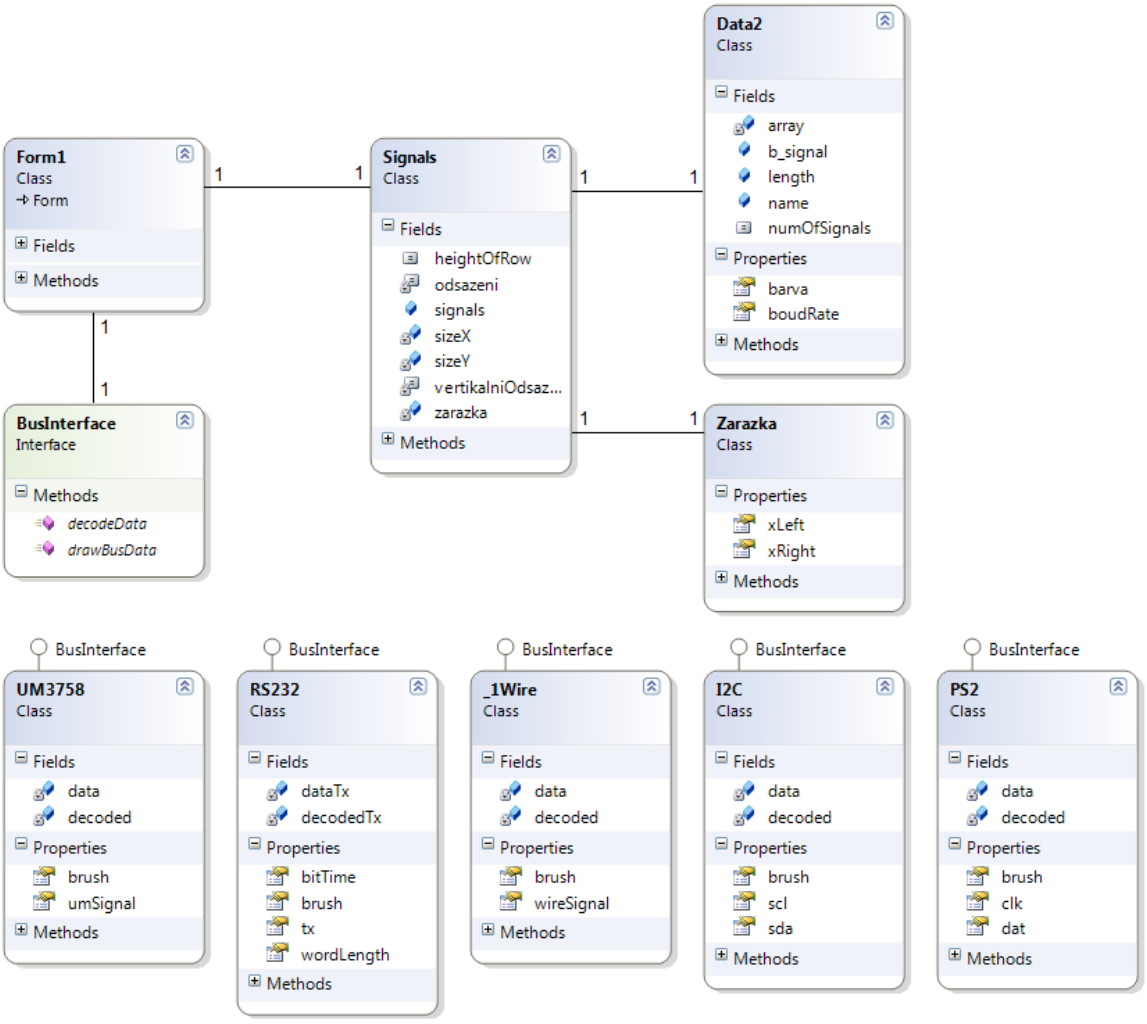
Po úspěšném přidání sběrnice projde analyzátor data. Nad místy kde jsou rozpoznány rámce se vypíší jednotlivé bity a pro užitečná data se zobrazí hexadecimální hodnoty.

4.2 Návrh

Aplikace má tři základní oblasti: uchování dat, zpracování dat a uživatelské rozhraní. Každá z oblastí má svůj jmenný prostor, v němž jsou uloženy příslušné třídy.

Na Obrázku 17 lze vidět zjednodušený diagram tříd. Základem je třída Form1, která prakticky tvoří uživatelské rozhraní. V této třídě je instance třídy Signals, která má za úkol uchování dat. Ve třídě Data2 jsou uchovány vzorky a ve třídě Zarazka jsou uloženy pozice zarážek pro měření času. Druhá relace třídy Form1 vede do rozhraní BusInterface. Do této proměnné se po přidání sběrnice ukládá jedna ze tříd umožňující dekódování a vykreslování dat dané sběrnice.

Třídy dalších formulářů jsem z diagramu vynechal, protože nemají na pochopení a chování aplikace zásadní vliv.



OBRÁZEK 17: DIAGRAM TŘÍD

4.2.1 Uložení dat

Všechny třídy pro uchování dat jsou ve jmenném prostoru nData. Jádrem aplikace je třída Data2. V ní se uchovávají data signálů, jejich jména a nastavení.

Samotná data jsou uložena v poli celých čísel a to tak, že každé jedno číslo reprezentuje jeden vzorek pro všech osm signálů. Jednotlivé bity tohoto čísla v dvojkové soustavě pak reprezentují logické hodnoty jednotlivých linek s tím, že 2^0 je bit první linky, 2^1 je bit druhé linky a tak dále až po 2^7 což je bit poslední osmé linky.

Ostatní informace o signálech jsou uloženy v jednorozměrných polích o osmi prvcích. V poli stringů jsou jména, v poli boolean pak nastavení zda je signál zobrazen či nikoliv. Pro barvy je vytvořeno pole instancí třídy Color ze jmenného prostoru Systém.Drawing.

Třída Data2 je pak ještě obalena třídou Signals, ve které jsou uloženy informace důležité hlavně pro vykreslování. Například výška jednoho řádku určeného pro signál nebo šířka okna. Je zde také instance třídy Zarazka.

4.2.2 Zpracování dat

Všechny třídy pro zpracování dat, se nacházejí ve jmenném prostoru LibraryOfsignals. Pro každou jednotlivou sběrnici je zde jedna třída, která ji umí dekódovat. Aby bylo usnadněno pozdější zvětšování podpory pro další sběrnice, musí každá z těchto tříd implementovat rozhraní BusInterface.

V rozhraní Businterface jsou pouze dvě metody. Jedna z nich se jmenuje decodeData , přijímá jeden parametr a sice odkaz na instanci třídy Data2. Tato metoda projde všechny vzorky a zjistí z nich jaký byl skutečný bitový provoz na sběrnici. Rozpozná počátky a konce rámců, řídící data atd. Výsledek pak uloží do pole čísel pro pozdější potřeby. Druhou metodou je drawBusData, ta vykreslí do obrázku část dekódovaných dat, která je zrovna viditelná.

4.2.3 GUI

Celé uživatelské rozhraní tvoří v podstatě jeden hlavní Windows formulář. V něm jsou dostupné všechny funkce a nastavení. Pro načítání dat, přidávání sběrnice a některá další nastavení jsou vytvořeny další podpůrné dialogy. Ty jsou umístěny ve jmenném prostoru Dialogs.

4.3 Klíčové části implementace

V následujících odstavcích popíši některé důležité části programu.

4.3.1 Načítání souboru

Soubory s daty se zpracovávají řádek po řádku. Na začátku každého souboru je hlavička, takže nelze hned číst užitečná data. V hlavičce je mimo jiné uložen počet vzorků, které následují v datové části ten se načte a podle něj se vytvoří dostatečně dlouhé pole pro uložení dat. Až se přečte řádek s obsahem : „[Data]”, nastaví se příznak pro čtení užitečných dat. Od teď se každý načtený řádek vždy rozdělí na pole řetězců. Oddělovacím znakem mezi jednotlivými řetězci je tabulátor. Po té se

z požadovaných řetězců přečtou jednotlivé jedničky a nuly a ty se složí do celého čísla. Výsledkem tohoto součtu je jeden vzorek, který se uloží do pole s výsledky.

```
while ((s = sr.ReadLine()) != null)
{
    if (b)
    {
        buffer = s.Split(new char[] { '\t' });
            //z jednotlivych bitu se secte cely vzorek
        for (int i = 0; i < 8; i++)
        {
            pom += (Int32.Parse(buffer[7 - i])) << (7 - i);
        }
        this.array[index++] = pom;
        pom = 0;
    }
    else
    {
        if (s.Length > 14)
        {
            if ("Sample amount:".Equals(s.Substring(0, 14))) //ziska pocet
vzorku
            {
                buffer = s.Split(new char[] { ' ' });
                int length = Int32.Parse(buffer[2]);
                this.array = new int[length];
                this.length = length;
            }
        }
        if ("[Data]".Equals(s)) //nastavi priznak ze uz nasleduji data
        {
            b = true;
        }
    }
}
```

VÝPIS 1: UKÁZKA CYKLU PRO NAČÍTÁNÍ DAT ZE SOUBORU

Pokud proběhne celý proces načítání vpořádku vrací metoda hodnotu 1, v opačném případě vrací -1.

4.3.2 Dekódování dat

Dekódování probíhá v podstatě u všech sběrnic velice podobně. V cyklu se procházejí jednotlivé vzorky a sledují se linky, jež byly nastaveny jako důležité. Protože komunikace vždy probíhá po nějakém druhu rámce nachází se sběrnice v různých pomyslných stavech. Proto je v cyklu větvení a podle toho jaké chování se zrovna od sběrnice očekává, takový kus kódu se provede. Základním stavem je klid na sběrnici. V tomto stavu se očekává nějaká forma start bitu. Až se podmínka start bitu splní, aktualizuje se stav na následující, například čtení adresy. Po tom co je adresa načtena znovu se změní stav na následující. Tímto postupem se data čtou a stav postupně prochází celým formátem rámce až po konečný stop bit. Přibližný postup jak vše probíhá jde vidět ve Výpisu 2.

```
for (int i = 0; i < signals.length; i++)
{
    bit = signals.getBit(tx, i);
    switch (status)
    {
        case 0:          //sbernice je v klidovem stavu
            if (bit == 0)
            {
                if (startBit)
                {
                    status = 1;
                    data[i] = 100;      //start bit - 100
                }
            }
            break;
        case 1:          //ctou se data
            if (data)
            {
                data[i] = 200 + bit;    //200 + x - datove bity
            }else
            {
                  status = 2;
            }
            break;
        case 2:         //cte se parita
            if (parita)
            {
                data[i] = 300 + bit;    //300 + x - paritni bity
            } else
            {
                  status = 3;
            }
            break;
        case 3:     //precte se stop bit
            if (stop)
            {
                data[i] = 400;    //400 stop bit
                status = 0;
            }
            break;
    }
}
```

VÝPIS 2: ZJEDNODUŠENÝ CYKLUS DEKÓDOVÁNÍ SBĚRNICE

Jak lze z kódu vidět, ukládají se do pole s daty pomocné hodnoty a ne pouze jedničky a nuly. Díky tomu pak lze v datech jednodušeji hledat a je okamžitě poznat v které fáze přenosu je daný bit součástí.

4.3.3 Vykreslování výsledků

Výsledek získaný dekódováním chceme samozřejmě zobrazit uživateli v okně s průběhy signálů. Pro tento účel je v horní části panelu ponecháno místo. Zobrazené výsledky se skládají ze dvou částí. Jednak jsou vypsány jednotlivé bity jako start, acknowledge, bity dat atd. Pod těmito údaji jsou zobrazeny rámce s vepsanými hodnotami nesených dat.

Protože při změně viditelných dat se musí vše vykreslovat znovu, bylo by velice neefektivní pokaždé rámce získávat znovu. Proto se po prvním dekódování spouští ještě druhá fáze. V ní se projdou už zjištěná data a sestaví se do rámců. Každá část je pak tvořena trojicí čísel. První a třetí číslo udávají hraniční vzorky, mezi nimiž se daný fragment nachází. Prostřední číslo udává hodnotu. Takže když je nutno zobrazit určitou část dat, porovnají se akorát počáteční a konečná čísla fragmentů s hranicemi viditelné oblasti. Jsou-li hraniční čísla fragmentu uvnitř rozsahu zobrazených dat dojde k vykreslení jinak se nic neděje.

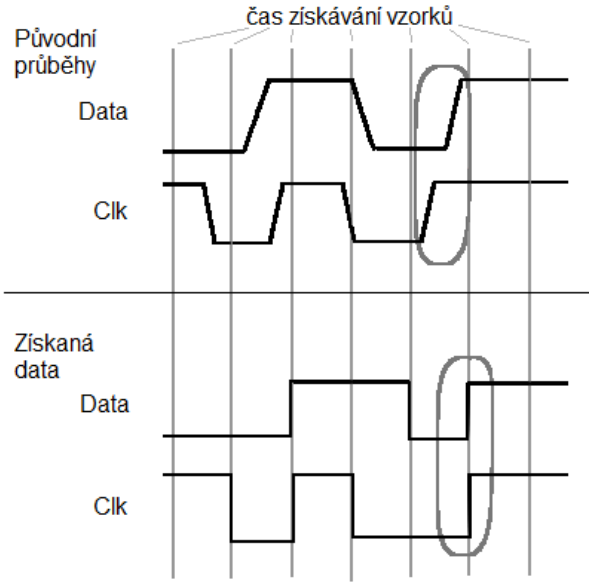
5. Testování

Důležitou součástí vývoje je testování všech funkcí a odladňování chyb. V tomto případě byl největší důraz kladen na dekódování sběrnic.

Testování sběrnic bylo poměrně problematické, protože k některým např. 1-Wire™ se špatně sháněla testovací data. Další překážkou je obrovské množství různých problémů, které se mohou v reálném provozu objevit. Pokud jde vše dobře a sběrnice pracuje jak má, pak není problém ani v jejím dekódování. Ovšem těžko si lze představit důvod proč analyzovat sběrnici, která funguje správně. Důvodem k použití logického analyzátoru je právě hledání chyb v přenosech. Zde vychází najevo první problém.

Dekódování sběrnic funguje na principu pamatování různých stavů a očekávání příslušné odezvy, jak bylo popsáno již dříve v této práci. Proto při výskytu chyby se tato šíří stále dál a nakonec i rámce které mohly být původně v pořádku se zobrazí špatně.

Dalším problémem, který ovšem ani moc nezávisí na softwaru je problém nedostatečné vzorkovací rychlosti. S tímto jsem se při testování potýkal poměrně často. Nedostatečná frekvence získávání vzorků může způsobit, že i rámce které byly původně v pořádku se nakonec jeví jako poškozené.



OBRÁZEK 18: UKÁZKA ZTRÁTY INFORMACE VLIVEM ŠPATNÉHO VZORKOVÁNÍ

Na Obrázku 16 lze vidět příklad chyby způsobené nedostatečnou rychlostí vzorkování. Konkrétně jde o přeskočení příznaku stop u sběrnice I2C. Přestože předchozí data se všechna zachytila správně, na konci rámce došlo ke ztrátě informace. Tím pádem jsou tato data ještě správná, ale chyba se šíří dále a následující rámec již bude znehodnocen.

6. Závěr

V rámci této práce byl položen základ softwaru pro analýzu sériových protokolů. Většinu předsevzetých cílů se podařilo splnit. Přesto je ponechán ještě poměrně velký prostor pro zdokonalení.

Jednou z možných oblastí pro vylepšení je rozšíření podporovaných rozhraní o některá další často používaná jako je SPI.Upravit lze také uživatelské rozhraní, hlavně přidání možnosti změnit velikost okna. Tak aby uživatelé s větším monitorem mohli plně využít jeho potenciál a tím si usnadnili práci.

Šíření chyby při dekódování, které jsem popsal v minulé kapitole je poměrně velkou nepříjemností při dekódování sběrnic. Jednou z možností jak tento problém vyřešit či minimalizovat jeho dopad by mohlo být přidání funkce nastavení počátku dekódování na určený vzorek. Potom v případě problému, pokud by uživatele zajímala i data za vyskytlou chybou, mohl by si jednoduše nastavit počátek dekódování až za poškozený rámec.

Zajímavou funkcí k implementaci by také mohla být možnost upravovat data přímo v obrázku. Takže při výskytu drobných chyb, které může způsobit třeba nedostatečná frekvence vzorkování nebo technické nedostatky sondy. By se tyto chyby daly dodatečně odstranit.

Literatura

[1] van Dorsten, Arian, *A logic analyzer using the PC's parallel port*, http://www.xs4all.nl/~jwasys/old/diy2.html, 2010

[2] CWAV Inc., *USBee PC and USB Oscilloscope, Logic Analyzer, Signal Generator, Protocol Decoder and Analyzer*, http://www.usbee.com/suite.html, 2010

[3] Dudáček, K., *Sériová rozhraní SPI, Microwire, I2C a CAN,* http://home.zcu.cz/~dudacek/NMS/Seriova_rozhrani.pdf, 2002

[4] Tišnovský, Pavel, *Sériový port RS-232C*, http://www.root.cz/clanky/seriovy-port-rs-232c/, 2010

[5] Tišnovský, Pavel, *Komunikace pomocí sériového portu RS-232C*, http://www.root.cz/clanky/komunikace-pomoci-serioveho-portu-rs-232c/, 2010

[6] Tišnovský, Pavel, *Komunikace pomocí sériového portu RS-232C podruhé*, http://www.root.cz/clanky/komunikace-pomoci-serioveho-portu-rs-232c-podruhe/, 2010

[7] Redakce HW serveru, *Sběrnice 1-Wire™*, http://hw.cz/Rozhrani/ART1215-Sbernice-1-Wire™.html, 2010

[8] Redakce HW serveru, *Stručný popis sběrnice I2C a její praktické využití k připojení externí eeprom 24LC256 k mikrokontroléru PIC16F877*, http://hw.cz/design/i2c_pic/index.html, 2010

[9] Chapweske, Adam, *The PS/2 Mouse/Keyboard Protocol*, http://www.computer-engineering.org/ps2protocol/, 2010

[10] Vašíček, *Zdeněk, Firmware / Řadič PS/2*, http://merlin.fit.vutbr.cz/FITkit/docs/firmware/fpga_ps2.html, 2010

[11] United Microelectronic Corp, *Tri-State Programmable Encoder/Decoder*, http://www.datasheetcatalog.org/datasheet/UnitedMicroelectronics/mXurzzt.pdf

[12] Intronix, *Intronix 34 Channel 500MHz PC based logic analyzer*, http://www.pctestinstruments.com/index.htm, 2010

Seznam obrázků

Seznam zdrojových kódů

Obsah CD

Adresář	**Popis**
Software	implementovaný software + testovací data
Text	text bakalářské práce
Zdroje	použité elektronické zdroje